Variable-resolution Compression of Vector Data

Yang, Bisheng ; Purves, Ross S ; Weibel, Robert

Abstract: The compression of spatial data is a promising solution to reduce the space of data storage and to decrease the transmission time of spatial data over the Internet. This paper proposes a new method for variable-resolution compression of vector data. Three key steps are encompassed in the proposed method, namely, the simplification of vector data via the elimination of vertices, the compression of removed vertices, and the decoding of the compressed vector data. The proposed compression method was implemented and applied to compress vector data to investigate its performance in terms of the compression ratio, distortions of geometric shapes. The results show that the proposed method provides a feasible and efficient solution for the compression of vector data, is able to achieve good compression ratios and maintains the main shape characteristics of the spatial objects within the compressed vector data.

DOI: https://doi.org/10.1007/s10707-007-0036-x

Geoinformatica (2008) 12:357–376
DOI 10.1007/s10707-007-0036-x

Variable-resolution Compression of Vector Data

Bisheng Yang · Ross S. Purves · Robert Weibel

Received: 28 January 2007 / Revised: 15 August 2007 /
Accepted: 27 August 2007 / Published online: 4 October 2007

Abstract The compression of spatial data is a promising solution to reduce the space of data storage and to decrease the transmission time of spatial data over the Internet. This paper proposes a new method for variable-resolution compression of vector data. Three key steps are encompassed in the proposed method, namely, the simplification of vector data via the elimination of vertices, the compression of removed vertices, and the decoding of the compressed vector data. The proposed compression method was implemented and applied to compress vector data to investigate its performance in terms of the compression ratio, distortions of geometric shapes. The results show that the proposed method provides a feasible and efficient solution for the compression of vector data, is able to achieve good compression ratios and maintains the main shape characteristics of the spatial objects within the compressed vector data.

Keywords variable-resolution · spatial data compression · error evaluation

1 Introduction

Vector map data are widely used in Mobile Geographical Information Systems (Mobile GIS), Location Based Services (LBS), mobile computing/query, vehicle navigation, web mapping, and so on. The storage of vector data, particularly of large scales, such as are typically appropriate in applications with a very local outlook, has a large impact on both transmission, rendering and response times especially for small devices with low bandwidth and limited processing power. Typical vector data volumes encountered in detailed vector data sets are therefore likely to prove difficult to handle on mobile devices.

B. Yang (✉)
State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing,
Wuhan University, Wuhan 430079, China
e-mail: bshyang@whu.edu.cn

B. Yang · R. S. Purves · R. Weibel
Department of Geography, University of Zurich,
Winterthurerstr. 190, CH-8057 Zurich, Switzerland

Data compression is a common approach to both decreasing the storage space required for spatial data and to minimizing the resultant transmission times. Various sophisticated algorithms have been implemented for the compression of raster data (e.g., remote sensing imagery, DEMs) such as wavelet-based compression algorithms (e.g., [1]), and for the compression of triangulated irregular network (TIN) based models (e.g., terrains). De Floriani et al. [3] designed compact data structures for compressing the connectivity of TIN based models. Park et al. [11] implemented an algorithm to compress TINs based on Delaunnay triangulation. Valette and Prost [15] proposed a progressive compression scheme for TINs based on the transformation of wavelets. Nevertheless, algorithms for compressing raster data and TIN based models can not directly deal with the compression of vector data because of the intrinsic complexity of topology within vector data.

During recent decades, various map generalization algorithms have been developed to derive small scale maps from large scale source data (e.g., [16]). Map generalization algorithms, by simplifying the representation of cartographic data, also reduce the data storage requirements in the generalized data. However, fully automated map generalization is still some way from being achieved [16] and a discussion of map generalization algorithms is beyond the scope of this paper. Nonetheless, by observing the obvious truth that generalized data volumes are smaller, it is possible to pose the question as to how we might reduce data volumes, not by cartographically generalizing but rather by removing extraneous detail which is not relevant to specific applications.

One potential approach to this problem is through the compression of vector data. The aims of vector data compression methods are to:

- remove redundant geometric data;
- maintain the main shape characteristics of objects; and
- preserve consistent topology within the compressed results.

Data compression can be considered to be lossy or lossless [9]. Lossless compression methods (e.g., Huffman coding, arithmetic coding) first detect the probability of each symbol, and then encode them according to some coding algorithms and require a decoding algorithm to reconstruct the original data from the compressed data. The reconstructed data is identical to the original. For example, Huffman coding based algorithms create a Huffman tree according to the frequency or probability of a symbol's appearance [5] and encode high probability symbols with low bits. Arithmetic coding based algorithms represent each possible sequence of n symbols by a separate interval on the number line between 0 and 1 [9]. Vector quantization (VQ) is a lossy data compression method based on the principle of block coding. It is a fixed-to-fixed length algorithm [4]. The main principle of VQ is as follows. Given a vector source with its known statistical properties, a distortion measure and the number of vectors, find a dictionary (codebook) that results in the smallest average distortion. Therefore, the procedure of VQ is to create a dictionary with S' symbols from the message with S symbols ($S'<S$) and to build a mapping between S' and S. VQ is widely used in the compression of images and video. Other lossy compression methods include wavelet-based compression, which, however, would be likely to have difficulties maintaining consistent topology within a set of lines or polygons. Shekhar et al. [14] proposed a cluster-based method for the compression of vector data. The method constructs a dictionary to compress the coordinate differences of adjacent vertices in polylines according to *K-means* method [7] and FHM methods [13] respectively, and decodes the original vector data based on the constructed dictionary. The FHM method requires the sizes of a set of nested squares for the construction of a dictionary. It is clear that the sizes

of the nested squares have an important effect on the accuracy of the compressed results. On the other hand, the K-means method follows a simple method to classify a given data set through some selected k clusters fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids should be placed in an intelligent way because different locations cause different results. As a result, the dictionary constructed by *K-means* method is very sensitive to the prerequisite conditions (e.g., the number of clusters, the initial positions of clusters), which might make the accuracy of the compressed results difficult to control. Secondly, the method of Shekhar et al. [14] may have difficulties in maintaining consistent topology within the compressed results. For example, suppose that one vertex is shared by two lines. The shared vertex may have different coordinates in the two compressed lines as the method of Shekhar et al. [14] compresses the coordinate differences of adjacent vertices without considering topological relationships within objects. On the other hand, a lot of algorithms were developed to measure polygonal approximation based on dynamic programming [6, 10, 12]. These algorithms aim at finding a subset of the original vertices to represent a polygonal curve and maintaining an optimal approximation of the polygonal curve. These algorithms can be applied to compress polygonal curves. However, they might have difficulties to maintain topological consistency of a set of polygonal curves (e.g., vector map data) as the topology between the polygonal curves might change after compression.

This paper proposes a new method for the variable-resolution compression of vector data with error control. The proposed method is a lossy compression method and aims to compress vector data while preserving consistent topology. Following the introduction, the framework and the implementation of the multi-resolution compression method are presented in the "2" section. The "3" section presents a set of methods to evaluate the quality of the compressed results in terms of the positional errors of the vertices and measurement of distortions in individual objects. The "4" section illustrates the experimental results before a discussion of the method's possibilities is set out at the end of the paper.

2 Framework of the variable-resolution compression method

The method presented here aims to reduce the volume of vector data transmitted through compression, whilst simultaneously maintaining the main geometric characteristics of the data and attempting to preserve topology. The method described is intended for use on spaghetti data (that is without any explicit topology) representing lines and polygons. Datasets containing points can also be processed, but the points themselves will not be modified by the algorithm. It is assumed that within the spaghetti data each object O_i consists of a set of vertices V_j where i is a unique id referring to the object and j is an ordered vertex. Furthermore, a table storing the degree of each vertex must also be created, together with the objects with which it is associated.

Our compression method can be broadly described as follows:

- Identification of candidate vertices for removal by applying the rules of Yang et al. [18]—these rules prevent topological inconsistencies from occurring when vertices are removed from the original geometry.
- Calculation of the displacement of removed vertices from simplified geometry and storage in a table together with their associated object ID i and position j in the original object. The identification of candidate vertices for removal and calculation of their displacements is described in the "2.1" section.

- Based on the displacements calculated, removed vertices are assigned to a TIN and then clustered according to their displacement from the original geometry.
- Entries in the table describing vertex displacements are replaced with a single reference to the cluster to which they belong. Generation of the TIN, clustering and calculation of the dictionary are described in the "2.2" section.
- Finally, a decoding method is implemented on the client side to decompress the compressed data and described in the "2.3" section.

2.1 Identifying and ranking candidate vertices for removal

The first step in the compression method is to identify potential vertices for removal. In order to meet our joint aims of preserving geometry and topology, we apply the method of Yang et al. [18] to select and rank vertices for removal. The method as implemented was designed for progressive transmission of vector data, a subtly different problem where the client receives first a skeleton of the data, then increasing detail until either the user terminates downloading or the complete original dataset is delivered. The basic procedure is as follows:

1. For every O_i, calculate the triangle area formed by every V_{j-1}, V_j, V_{j+1}
2. Rank the resulting vertices for removal according to increasing triangle area
3. For every vertex, check whether it may be flagged for removal or not, according to the following rules :

   (a) Start and end vertices of line objects may not be removed
   (b) If a vertex is the only vertex shared by two objects it may not be removed
   (c) If a vertex is shared by more than two objects it may not be removed
   (d) If vertex removal results in intersection between objects where none previously existed, it may not be removed

The first three rules are easily tested by checking the table associated with the original geometry. However, the final rule requires that new geometry can be calculated and tested for self-intersection or changes in the topological relationships between objects. To maximize the efficiency of these checks the bounding box of the segment associated with the removed vertex (i.e. the segment formed by V_{j-1}, V_{j+1}) is calculated and only the topology of objects lying within this region checked.

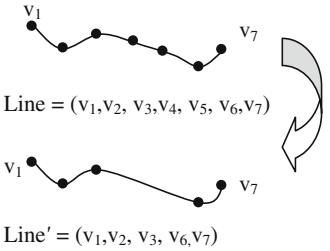
The result of this process is a set of vertices W_k whose removal will not change the topology of the original dataset ranked according to their order of removal and a set of irremovable vertices X_l. which form a skeleton of the original geometry ($X_l \notin W_k$; $W_k \in V_j$; $X_l \in V_j$) This process is identical to that described in Yang et al. [18], with the exception that independent objects of some given minimum size are not removed.

Having identified and ranked available vertices for removal, a table is constructed in which are stored the vertices W_k, the position j of the vertex in the original geometry, and the object i with the vertex is associated. For each vertex W_k the displacement of the vertex from the original geometry is calculated as follows and stored in the table as illustrated in Fig. 1:

$$\Delta x_k = x_j - \left(x_{j-1} + x_{j+1}\right)/2 \tag{1}$$

$$\Delta y_k = y_j - \left(y_{j-1} + y_{j+1}\right)/2 \tag{2}$$

where $V_j \in O_i$ and $V_j = V_k$



remove order	displacement of removed vertices	object id	position in associated object
No.1	$\Delta x_4, \Delta y_4$	1	4
No.2	$\Delta x_5, \Delta y_5$	1	5

Fig. 1 The storage of the displacement of removed vertices during removing vertices

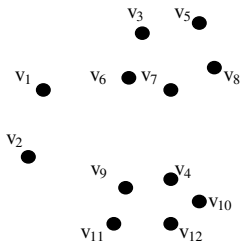
2.2 Compressing the vertex displacements

We compress the data by replacing the individual entries referring to displaced vertices in the table described above (Fig. 1) with a dictionary in which references to displaced vertices are replaced with a single reference to a cluster describing "average" vertex displacement. Many existing algorithms, for example K-means, could be applied to form clusters but most existing methods require some parameters which cannot be easily related to geometric constraints (i.e. the number of clusters required or initial cluster positions). We therefore propose using a clustering method based on a geometric criterion—the distance of the individual vertex displacements (Δx_k, Δy_k) from one another. The clustering algorithm allocates vertices to the dictionary as follows:

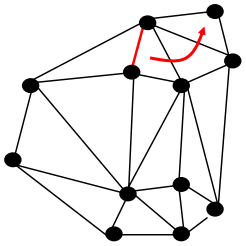
- A TIN is generated, where each vertex represents a vertex displacement in the table and TIN edges therefore represent the difference in displacement between edges
- TIN vertexes are allocated to clusters, where all vertices within a threshold distance of one another are allocated to the same cluster
- Clustered vertices are then allocated an entry in the dictionary, where clustered vertices are represented by a single displacement vertex which minimizes the resulting positional errors
- Vertices which do not belong to any cluster are retained as individual values in the dictionary

The generation of the TIN is carried out using a dividing and conquering algorithm [2]. Having generated a TIN, some edge with a length of less than the threshold distance is identified as a seed point, and the two associated vertices are allocated to a cluster. The left and right triangles of the original edge are then traversed, and the distance of every new vertex to every vertex already in the cluster calculated. If this distance is less than the threshold distance, then the new vertex is added to the cluster, and the associated triangles recursively traversed until no further cluster members are identified. At this point, the cluster is complete, and the associated vertices and edges can be deleted from the TIN. The algorithm continues until only edges longer than the threshold distance remain, and therefore no further clusters can be identified. The remaining vertices are simply allocated to the dictionary.

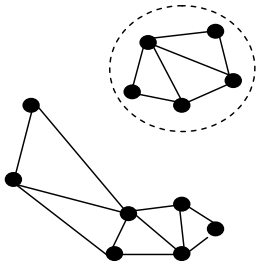
Finally, members of clusters are allocated a single displacement value which minimizes the positional error of the individual vertices by identifying a cluster vertex which minimizes the sum of the squared differences between the original displacement vertices and the cluster vertex. The positions of cluster vertices were decided by dynamic programming. The overall procedure of clustering generation is illustrated in Fig. 2.



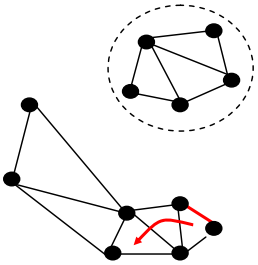
a displacement of removed vertices



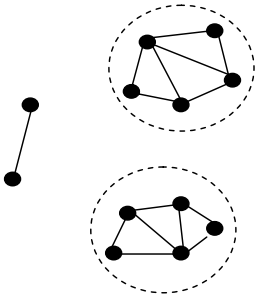
b triangles recursive traverse



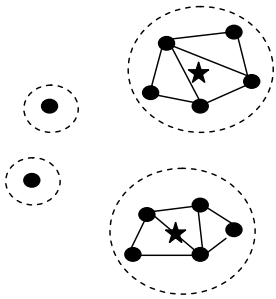
c clustering generation



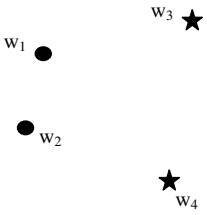
d triangles recursive traverse



e clustering generation



f cluster vertex generation



g generating clusters positions

entry id	entry	mapping
1	w_1	v_1
2	w_2	v_2
3	w_3	v_3, v_5, v_6, v_7, v_8
4	w_4	$v_4, v_9, v_{10}, v_{11}, v_{12}$

h dictionary generation

Fig. 2 Clustering generation and storage from the displacement of removed vertices according to a distance threshold

2.3 Decoding of the compressed vector data

To reconstruct the original polygons or polylines, we implement a decoder to decode the compressed vector data. Compressed vector data consists of a simplified version of the original vector data, a table of the coordinate increments, and a dictionary. The simplified version is represented by a simple spaghetti model consisting of vertices, lines and polygons, which can be considered to be vertices, open chains of vertices and closed chains of vertices respectively. Moreover, each line or polygon has a unique id. The implemented decoder reads the simplified version into memory before traversing the table of the coordinate increments row by row to retrieve the index in the dictionary, the positional

order in and the id of the associated polygon or polyline of the coordinate increment. Then, the decoder calculates the coordinates of the removed vertex according to the coordinate increment and its positional order and retrieves the corresponding polygons or polylines from the simplified version according to the id of polygon or polyline. Finally, the decoder inserts the removed vertex into the vertex chain of corresponding polygons or polylines at a specified position according to its positional order. Once all the rows in the table of the coordinate increments are traversed, the compressed vector data is decoded, as illustrated in Fig. 3.

3 Methods for evaluating positional errors and distortions

The proposed compression method decodes the vector data according to the constructed dictionary. As many entries in the dictionary are generated from clusters and therefore encompass positional errors, compressed lines and polygons will be distorted. In assessing whether the algorithm has met the aims set out in the introduction, that is to say to compress vector data whilst maintaining shape characteristics and preserving topology, it is possible to both qualitatively and quantitatively compare the original data with the decompressed data after the lossy algorithm has been applied.

The compressed vector data can be visually inspected to compare the similarity of geometric shapes and to check for topological problems (e.g., self-intersection). Furthermore, quantitative measurement can be employed to quantify the distortion of polylines and polygons. McMaster [8] proposed a set of methods to evaluate the distortion of simplified lines, namely, length, density, angularity, and curvilinearity. In this study, we measured length change of lines to evaluate the distortions of the compressed lines, expressed as Eq. 3.

$$L_{\text{distortion}} = \left(L_0 - L_0^{'}\right) \Big/ L_0 \tag{3}$$

where L_0 is the length of the original line, $L_0^{'}$ is the length of the compressed line.

As the implemented decoder decodes the compressed lines by retrieving all the removed vertices, the numbers of reconstructed lines is identical to that of the original dataset. Therefore, there are no changes in density and curvilinearity of the compressed lines. The measurement of area change of polygon is a feasible solution for the evaluation of distortion of polygon before and after simplification in cartographic generalization. We adopt Eq. 4 to measure the distortion of the compressed polygons, as illustrated in Fig. 4.

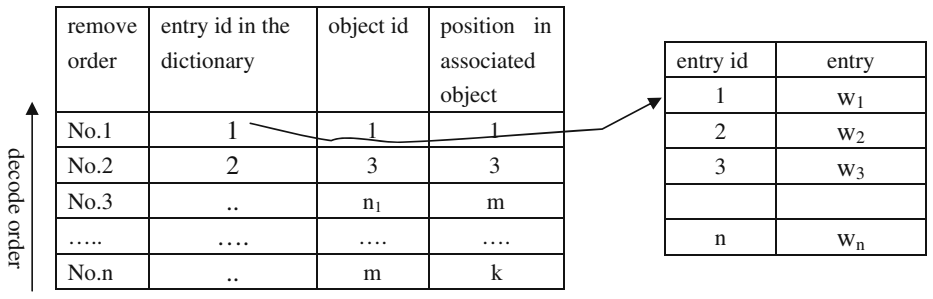


Fig. 3 Decoding of the compressed data according to the dictionary

According to Eq. 4, the smaller the value of $S_{\text{distortion}}$ the better quality of the compression result is.

$$S_{\text{distortion}} = S\left(A \cap \overline{B} + \overline{A} \cap B\right) / S(A) \tag{4}$$

where A and B denote the original polygon and compressed polygon respectively.

Secondly, positional errors of vertices are an important criterion to evaluate the distortions of the compressed polygons or polylines. We calculate the RMSE (root mean square error) of vertex positions to measure the positional distortions of the compressed vector data. In light of the procedure of constructing the dictionary, the larger the value of the distance threshold the higher the compression ratio that can be achieved and the larger the resultant RMSE value.

$$\begin{cases} X_{\text{RMSE}} = \sqrt{\dfrac{\sum_{i=1}^{n} |x_i' - x_i|^2}{n}} \\ Y_{\text{RMSE}} = \sqrt{\dfrac{\sum_{i=1}^{n} |y_i' - y_i|^2}{n}} \end{cases} \tag{5}$$

where n is the total number of vertices in a vector data, (x_i, y_i) is the original vertex, (x_i', y_i') is the vertex decoded by the entries of the dictionary.

4 Experimental study

In order to assess the compression method's success, we present here an experimental study for two datasets, a polygonal dataset representing administrative boundaries in Canada and a linear dataset representing a hydrographic dataset (Table 1).

The experimental datasets were compressed according to the methods described in the "2" section for a range of distance thresholds and decoded to allow assessment of the effectiveness of the algorithm. It should be noted that only steps in the algorithm before

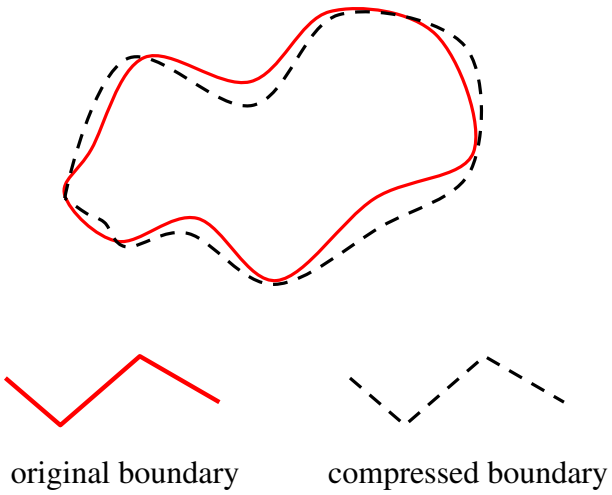


Fig. 4 The distortion measurement of compressed polygon

Table 1 Statistics of the experimental datasets

Data sets	The number of features	The number of vertices	Data volumes (kb)	Data set extent (km)	Nominal data precision (m)
Polygonal dataset	245	81,553	2,755	5,600×4,700	±100
Linear dataset	2,717	60,830	1,442	134×167	±24

clustering must be repeated for different distance thresholds—the identification of candidate vertices for removal, calculation of their displacements and generation of the TIN are independent of the distance threshold and need only be carried out once on a given dataset.

We present a number of quantitative and qualitative measures of the effectiveness of the algorithm, together with a comparison of the algorithm with two commonly used compression methods.

Our first criterion for effectiveness is based on the ratio of compression achieved, which as illustrated in the "3" section is a function of the distance threshold chosen (since with different distance thresholds the number of clusters stored in the dictionary is reduced). Compression performance for the two datasets is illustrated in Tables 2 and 3. In each case, we illustrate results for topologically acceptable cases and one value of a distance threshold (500 and 20 m for the polygonal and linear data respectively) where topological problems, as defined by importing the data into ArcMap and checking for self-intersections, occurred. The RMSE values for x and y coordinates for different value of the distance threshold are also shown in Tables 2 and 3. In both cases, most vertices are potential candidates for entry in the dictionary (in other words, they are not topologically important vertices)—98% of vertices in the case of the polygonal and 90% of vertices in the case of the linear data. However, the compression achieved is a function not of this initial step, but rather of the number of coordinate increments then stored in the dictionary. Thus, the total volume of the compressed dataset consists of the core set of vertices, the topological information associated with the removed vertices and the dictionary itself. Thus, for example with a distance threshold of 1 m the ratio of compression is 62% and the RMSEs of the x and y coordinates are 0.16 m. For a larger distance threshold of 5 m the ration of compression increases to 71.5% and the RMSEs of the x and y coordinates to about 7.5 m.

We measured the time costs of encoding and decoding of the proposed compression method based on the experimental datasets. In light of the steps of the proposed compression method, three steps are encompassed to encode a vector data, namely, simplification, generation of clusters and dictionary entries. Hence, we measured the time costs in the two steps respectively to calculate the overall time costs of encoding an original vector. To decode a compressed vector, the proposed compression method first reads the simplified version of an original vector and associated topology of coordinate increments into memory, then traverses the coordinate increments row by row in the dictionary. Hence, the time cost of decoding a compressed vector is the sum of time costs in the above two steps. The time of encoding and decoding of the proposed compression method was measured in a Laptop with 512 MB memory and Pentium IV CPU. Table 4 shows that the generation of clustering vertices occupies a large proportion time particularly a large distance threshold is set. For example, it takes 33.97 s to generate clusters with a distance threshold of 500 m and the overall encoding time is 37.21 s.

By comparing our method to "standard" compression algorithms, it is possible to get a sense of whether these compression ratios can be considered to be adequate. In Table 4 we illustrate compression ratios achieved by GZIP and arithmetic coding methods. Arithmetic coding, a lossless compression method, is a method of encoding data using a variable

Table 2 Compression performance (e.g. times of encoding and decoding, compression ratio) of the proposed compression method

A	The number of removed vertices (reduce by %)	Distance threshold (m)	The number of entries of dictionary (reduce by %) (data volume) (kb) *B*	Data volume of topology of removed vertices (kb) *C*	Data volume of simplified map (kb) *D*	Data volume of compressed map (kb) *E*	Ratio of compression (%)	RMSEs of coordinates	
								X (m)	*Y* (m)
Polygonal network (2,755)	80,136 (98)	1	58,340 (27) (455.78)	611.08	32.82	1,045.68	62.0	0.16	0.16
		5	44,766 (44) (349.73)			993.63	63.9	1.16	1.27
		10	25,180 (69) (196.72)			840.62	69.5	2.7	2.82
		25	18,096 (77) (141.38)			785.28	71.5	7.53	7.88
		500	8,417 (99) (65.75)			709.65	74.2	140.76	139.47
Linear dataset (1,442)	54,644 (89.8)	1	28,959 (47) (226.24)	320.17	118.62	665.03	54.9	0.31	0.21
		5	15,660 (71) (122.34)			561.13	61.1	1.67	1.70
		8	11,228 (79) (87.72)			526.51	63.5	2.87	2.90
		10	9,250 (83) (72.27)			511.06	64.6	3.71	3.67
		20	1,065 (99) (8.3)			477.09	67.0	3.94	3.91

Data volumes of compressed maps $E=B+C+D$. The ratio of compression=$(A-E)/A$.

Table 3 Compression performance (e.g. times of encoding and decoding, compression ratio) of the proposed compression method

Encoding			Decoding (s) (T_2)
Time of removing vertices (s) (t_0)	Time of clustering and dictionary generation (s) (t_1)	Encoding time costs (s) ($T_1=t_0+t_1$)	
3.24	3.73	6.97	34.04
	4.91	8.15	33.88
	6.61	9.85	33.68
	13.67	16.91	33.70
	33.97	37.21	33.92
1.12	2.92	4.04	0.45
	4.02	5.14	0.36
	6.19	7.31	0.75
	7.95	9.07	0.72
	21.25	22.37	0.79

number of bits. The number of bits used to encode each symbol varies according to the probability assigned to that symbol. Low probability symbols use many bits and high probability symbols use fewer bits. Both our method and arithmetic coding methods also require the implementation of a decoder on the client device. It is clear that our method achieves similar or better values of compression ratio in comparison to standard methods with, as will be discussed later, the added advantage that the displacement of removed vertices can be retrieved incrementally from our dictionary (this is not the case for arithmetic coding methods)

Our third set of measures of the effectiveness of the algorithms concern distortions in the original geometries. For topologically correct compressed datasets we measured the percentage changes in area for each polygon and of line length for a range of distance thresholds. Figure 5 shows box plots for the percentage distortions in polygon area and Fig. 6 for those in line length as a function of the distance threshold selected, where red line denotes the maximum the percentage distortions in polygon area and line length respectively. It can be seen from Fig. 5 that the area distortions of 98% of the polygons are less than 2%. Figure 6 shows that the length distortions of 90% of the polylines are less than 4%.

Table 4 The comparison of compression ratio

Dataset/data volume (kb)	Compression method	Compressed data volume (kb)	Compression ratio (%)
Polygonal dataset (2,755)	Gzip	1,168	57.6
	Arithmetic coding methods	1,252.42	54.5
	The proposed method (threshold=5 m)	993.63	63.9
Linear dataset (1,442)	Gzip	547.98	62.0
	Arithmetic coding methods	663.32	54.1
	The proposed method (threshold=5 m)	561.13	61.1

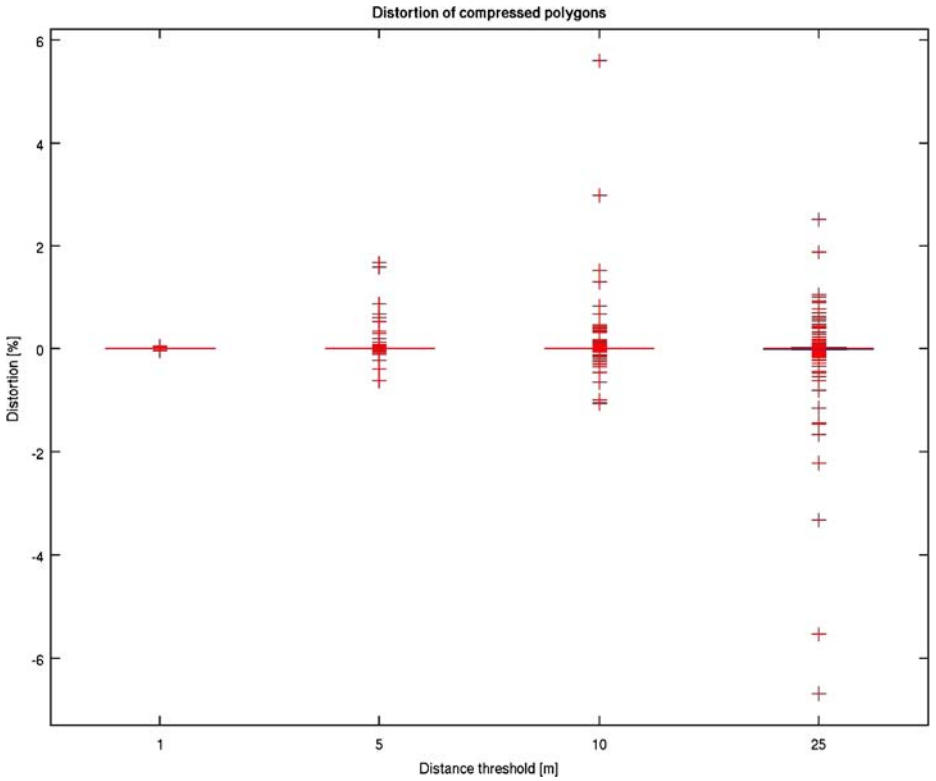


Fig. 5 The distortion of areas of compressed polygons with different distance thresholds

Finally, we present visualizations of the compressed data for qualitative inspection. Figure 7 show the original and compressed polygonal data with a distance threshold of 25 m, where effectively no differences are visible, and a zoomed in area of the figure to illustrate the magnitude of difference. Figure 8 illustrates the hydrological dataset with a distance threshold of 10 m.

5 Discussion

In the "4" section we presented a set of results to illustrate the effectiveness and limitations of our approach. Qualitative comparisons of the original data and decoded compressed data illustrate clearly that changes in the data are small for relatively large distance thresholds. Furthermore, we achieve values of compression ratio, which are as good or better than standard methods, with the added advantage that our method allows progressive decoding. Finally, the RMSEs of the reconstituted points are small (though it should be noted that this is only with respect to the vertices, which are reconstructed) and the distortions in shape are also limited as illustrated by the box plots in Figs. 5 and 6.

As stated above, a key advantage of our method is that it is possible to retrieve coordinate increments incrementally. This means that the decoder can be implemented to select according to the order of storage in the dictionary and thus data can be visualized and

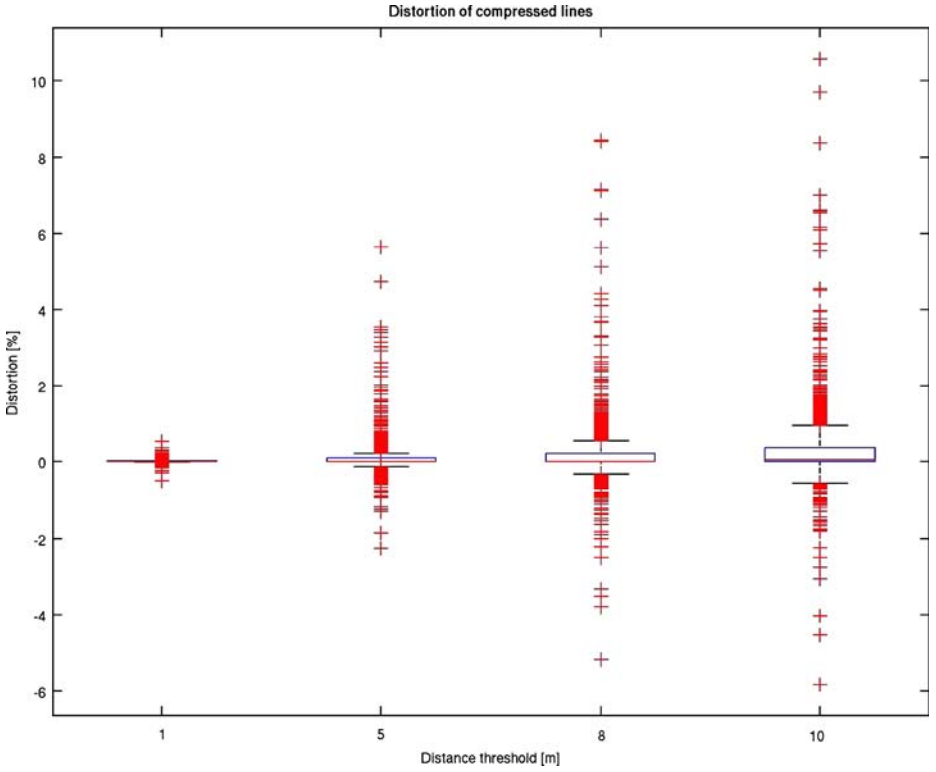


Fig. 6 The distortion of lengths of compressed polylines with different distance thresholds

queried even before the whole dataset is read into memory. This aspect of our approach is analogous to that of progressive vector transmission [18], but it is important to note the key difference—here we are delivering a dataset whose volume has been reduced through a lossy algorithm—that is to say we can never reconstruct the original geometry. On the other hand, the proposed method is able to compress vector data at variable resolutions when different distance thresholds are specified. However, the proposed method does not currently support spatial tiling, which might be necessary for mobile visualizations.

The use of a decoder also means that the real data volume associated with this compression method should include that associated with the decoder for one off downloads. However, the decoder is computationally simple, fast and could easily be integrated into a range of applications allowing, for example, mobile devices to more successfully deal with large volumes of vector data.

The key weakness in our method concerns the selection of a suitable distance threshold. This has three key implications. Firstly, the overall data compression ratio is dependent on the chosen distance threshold. Secondly, the distortion of the original data is a function of the distance threshold. For many applications, it is likely that not the mean, but the maximum (i.e. worst case) distortion should be used to select a suitable distance threshold. By playing off the compression ratio against distortion it is likely that suitable empirical thresholds can be generated. It is clear from the two examples given that the distance threshold is a function of the original dataset's precision, and by examining more datasets it

should be possible to make recommendations for distance thresholds for given dataset qualities. The third and most serious implication concerns the topological consistency of the dataset. The algorithm as implemented only allows detection of self-intersections during the decoding process. Once again these are a function of the distance threshold chosen, but as illustrated in Tables 2 and 3 they can have widely varying values for relatively similar data (500 and 25 m respectively). More empirical work is required to identify distance thresholds, which do not lead to self-intersection.

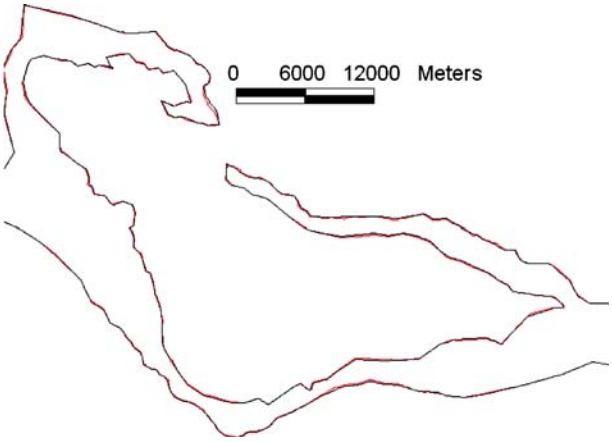
In the current study, we adopted a simple solution to overcome potential self-intersections. Suppose that a self-intersection occurs to a polygon or a polyline when an entry (x, y) in the dictionary is employed to decode the original polygon or polyline. The entry is generated from a cluster, in other words, it is not the original vertex of the polygon or the polyline. Therefore, the original vertex corresponding to the entry will be inserted into the dictionary as a new entry. The self-intersection will then be overcome. Our method can only check topology inconsistencies at the second step, namely the decoding of vector data. Therefore, the solution is feasible for the preservation of consistent topology only when the encoding and decoding of vector data is not executed on the fly. However, such a solution would be adequate for many compression tasks where the same compressed data are likely to be downloaded many times, since these checks need only be carried out once.

the original map

Fig. 7 The original map and compressed maps (polygonal dataset)

Fig. 7 (continued)

the compressed map (distance threshold=25m)



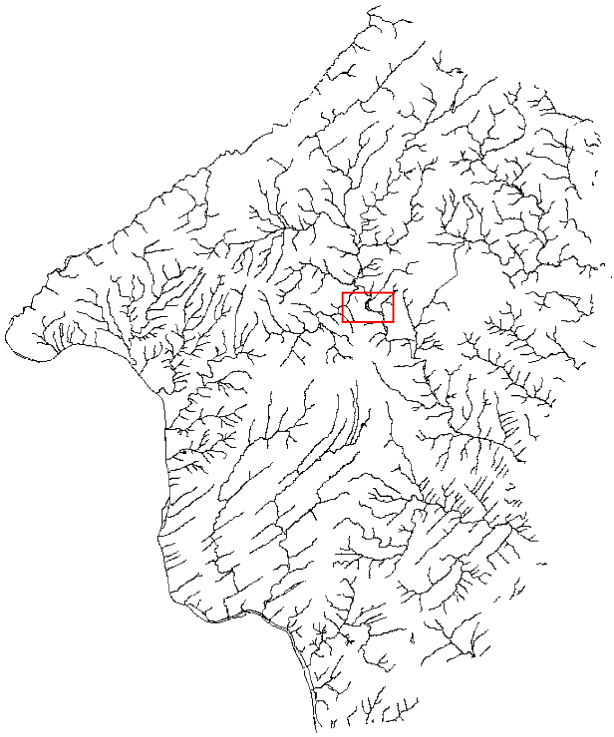
6 Conclusions

Vector data are widely used in LBS, vehicle navigation, web-mapping, and mobile visualizations. The compression of vector data is central to mobile visualizations, Internet transmission, and interactive visualizations because of increasing data volumes, narrow network bandwidths, and limited screen sizes [17]. In this paper, we propose and implement a new method to progressively compress vector data.

Two datasets were selected to evaluate the performance of the proposed compression method in terms of the compression ratio, the positional errors, and the geometric distortion. The experimental results show that the proposed compression method is able to:

- preserve consistent topology within the compressed results for a correctly selected distance threshold;
- compress vector data at different compression ratios according to varied requirements; and
- achieve a reasonable balance between the compression ratio and the distortions of geometric shapes.

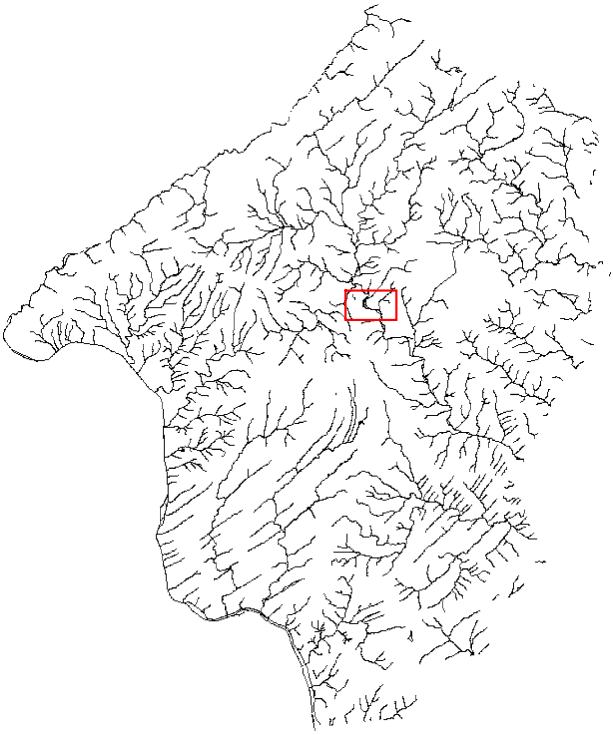
Further research will focus on developing recommendations for distance thresholds for different datasets and the integration of the compression method and the progressive transmission of spatial data for the improvement of transmission efficiencies of spatial data in web environments. Secondly, further research will extend the methodology to support



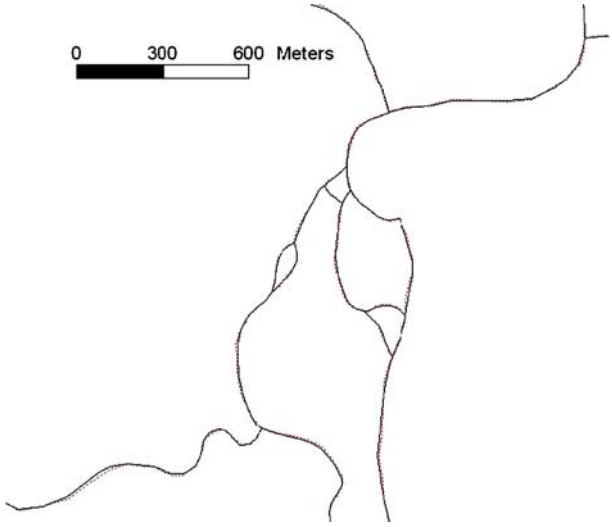
the original map

Fig. 8 The original map and compressed maps (polyline dataset)

Fig. 8 (continued)



the compressed map (distance threshold=10m)



spatial tiling/segmentation, which is necessary for some applications (e.g., mobile visualizations).

Acknowledgments This research was supported by projects from the National Natural Science Foundation of China (No. 40401051, No. 40571134), and the APTS project from the Research Council of the University of Zurich.

References

1. J.T. Bjørke and S. Nilsen. "Wavelets applied to simplification of digital terrain models," *International Journal of Geographical Information Science*, Vol. 17(7):601–621, 2003.
2. R.A. Dawyer. "A fast divide-and-conquer algorithm for constructing Delaunay triangulations," *Algorithmica*, Vol. 2(2):137–151, 1987.
3. L. De Floriani, P. Magillo, and E. Puppo. "VARIANT: A system for terrain modeling at variable resolution," *GeoInformatika*, Vol. 4(3):287–315, 2000.
4. A. Gersho and R.M. Gray. Vector Quantization and Signal Compression. 1st edition, Kluwer: Norwell, MA, 1991.
5. D.A. Huffman. "A method for the construction of minimum redundancy codes", *Proceedings of the IRE*, Vol. 40:1098–1101, 1952.
6. A. Kolesnikov and P. Franti. "Polygonal approximation of closed discrete curves," *Pattern Recognition*, Vol. 40:1282–1293, 2007.
7. J.B. MacQueen. "Some methods for classification and analysis of multivariate observations", in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297, University of California Press, Berkeley, 1967.
8. R.B. McMaster. "A statistical analysis of mathematical measures of linear simplification," *American Cartographer*, Vol. 13(2):103–116, 1986.
9. M. Nelson. The Data Compression Book. M&T Books, 1991.
10. J.C. Perez and E. Vidal. "Optimum polygonal approximation of digitized curves," *Pattern Recognition Letters*, Vol. 15:743–750, 1994.
11. D. Park, H. Cho, and Y. Kim. "A TIN compression method using Delaunay triangulation," *International Journal of Geographical Information Science*, Vol. 15(3):255–270, 2001.
12. M. Salotti. "Improvement of Perez and Vidal algorithm for the decomposition of digitized curves into line segments," *Proceedings of the 15th International Conference on Pattern Recognition*, Vol. 2:878–882, 2000.
13. D. Salomon. Data Compression: The Complete Reference. 2nd edition, Springer: Berlin Heidelberg New York, 2000.
14. S. Shekhar, Y. Huang, J. Djugash, and C. Zhou. "Vector map compression: a clustering approach," in *Proceedings of 10th ACM International Symposium Advances in Geographic Information Systems-GIS'02*, November 8–9, 2002, McLean, Virginia, USA, pp. 74–80, 2002.
15. S. Valette and R. Prost. "Wavelet-based progressive compression scheme for triangle meshes: Wavemesh," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 10(2):123–129, 2004.
16. R. Weibel and G. Dutton. "Generalizing spatial data and dealing with multiple representations," in P. Longley, M.F. Goodchild, D.J. Maguire, and D.W. Rhind (Eds.), *Geographical Information Systems: Principles, Techniques, Management and Applications*. 2nd edition (abridged edition), Wiley, Hoboken, NJ, 125–155, 2005.
17. B.S. Yang. "A multi-resolution model of vector data for rapid transmission over the internet," *Computers & Geosciences*, Vol. 31(5):569–578, 2005.
18. B.S. Yang, R.S. Purves, and R. Weibel. "Efficient transmission of vector data over the internet," *International Journal of Geographical Information Science*, Vol. 21(2):215–237, 2007.

BISHENG YANG obtained his Ph.D. degree in Photogrammetry and Remote Sensing from Wuhan University, China in 2002. He was a Post-doctoral Research Fellow in GIS Division, Department of Geography, University of Zurich, Switzerland from October 2002 to October 2006. He is now an Associate Professor in GeoInformatics at the State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, China. His research interests cover 3D data modelling, 3D GIS, 2D/3D visualization, LBS, progressive transmission of spatial data over the Internet. He has published a lot of scientific papers in IJGIS, IJRS, PE&RS.

ROSS PURVES is a lecturer in the GIS Division of the Department of Geography at the University of Zurich. Prior to coming to Zurich, he worked in the Geography Department of the University of Edinburgh. His research in GIScience covers a range of themes, from Geographic Information Retrieval to the environmental modelling of processes related to snow and ice and, in particular, issues relating to the uncertainty in modelling of such processes.

ROBERT WEIBEL is a professor of Geographical Information Science at the Department of Geography, University of Zurich. He obtained an M.Sc. and Ph.D. degrees in Geography with specialization in GIScience from the University of Zurich in 1985 and 1989, respectively. Before starting his academic career he was a software engineer in the GIS industry. He has served as chair of the Commission on Map Generalization of the International Cartographic Association (ICA, 1992–2003) as well as on numerous journal editorial boards and program committees of scientific meetings and conferences in GIScience. His current research interests are in map generalization and multiple representations, location-based services, and spatio-temporal analysis of moving point objects.